

## Real-time CORBA Becomes a Reality

### Contact:

Chris Vander Rhodes  
Object Management Group  
Phone: +1-508-820 4300 ext. 115  
Email: [chrisvr@omg.org](mailto:chrisvr@omg.org)

---

**Framingham, MA, USA— July 20, 1999--** The Object Management Group (OMG) just completed a vote to adopt the Real-time CORBA 1.0 specification. Traditionally, one of CORBA's strengths has been that it offers developers the ability to work at a high level of abstraction. This allows programmers to focus on the application rather than the minutiae of network programming, leaving detailed decisions on priority and task scheduling to the CORBA implementation. However, in some critical applications, the programmer needs to be able to override these defaults to exert more control over these lower level details.

Real-time CORBA offers a response to this need by allowing specialists to concentrate on the details and performance issues of working in both homogeneous and heterogeneous distributed environments. This approach builds on capabilities provided by existing real-time operating systems; it does not try to duplicate or replace those capabilities. It establishes a fundamental set of broadly applicable real-time CORBA capabilities, based on mature technologies, and is not only usable on its own merits, but also provides a stable basis for extension.

While this need was initially recognized in specialized environments, such as Air Traffic Control, the real-time extensions to CORBA's ability are very important to other industries that also require control over Quality of Service and end-to-end predictability. This is a common trend in the development of OMG specifications. While specifications are often developed to meet the needs of a very particular group, the OMG's open process encourages input and adoption of those specifications to meet wider industry needs.

Professor Doug Schmidt of Washington University, a recognized expert in the area of real-time systems, said, "There has been a surge in demand for standards-based real-time middleware in a many domains including telecom, aerospace, process control, and financial services. The OMG's Real-time CORBA specification is the first middleware standard to meet this demand head on."

One of the most promising markets for the application of Real-time CORBA is in the telecommunications industry. Lucent Technologies plans to develop a product using Real-time CORBA, which is expected to be available in beta in September 1999. Besides Lucent, other submitters on the Real-time 1.0 specification include Alcatel Alsthom Recherche, Hewlett-Packard, Highlander Communications, IONA, Borland Corporation, Lockheed Martin, Nortel, Object-Oriented Concepts, Objective Interface Systems, Sun Microsystems and Tri-Pacific Software.

###

CORBA®, The Information Brokerage®, CORBA Academy®, and the Object Management Group logo® are registered trademarks of the Object Management Group. OMG™, Object Management Group™, the CORBA Logo™, ORB™, Object Request Broker™, the CORBA Academy logo™ IIOP™, XMI™, MOF™, OMG Interface Definition Language™, IDL™, CORBAservices™, CORBAfacilities™, CORBAMED™, CORBANet™, UML®, the UML Cube Logo™, and Unified Modeling Language™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

# Patterns and Performance of Real-time Object Request Brokers

Douglas C. Schmidt

Associate Professor  
schmidt@uci.edu  
www.ece.uci.edu/~schmidt/

Elec. & Comp. Eng. Dept.  
University of California, Irvine  
(949) 824-1901



## Sponsors

NSF, DARPA, ATD, BBN, Boeing, Cisco, Comverse, GDIS, Experian, Global MT, Hughes, Kodak, Kronos, Lockheed, Lucent, Microsoft, Mitre, Motorola, Nokia, Nortel, OCI, Oresis, OTI, QNX, Raytheon, SAIC, Siemens SCR, Siemens MED, Siemens ZT, Sprint, Telcordia, USENIX

High-performance, Real-time ORBs

Douglas C. Schmidt

## Motivation: the QoS-enabled Software Crisis



[www.arl.wustl.edu/arl/](http://www.arl.wustl.edu/arl/)

- Symptoms
  - Communication **hardware** gets smaller, faster, cheaper
  - Communication **software** gets larger, slower, more expensive
- Culprits
  - *Inherent* and *accidental* complexity
- Solution Approach
  - *Standards-based COTS Hardware & Software*

UC Irvine



1

High-performance, Real-time ORBs

Douglas C. Schmidt

## Problem: the COTS Hardware & Software Crisis



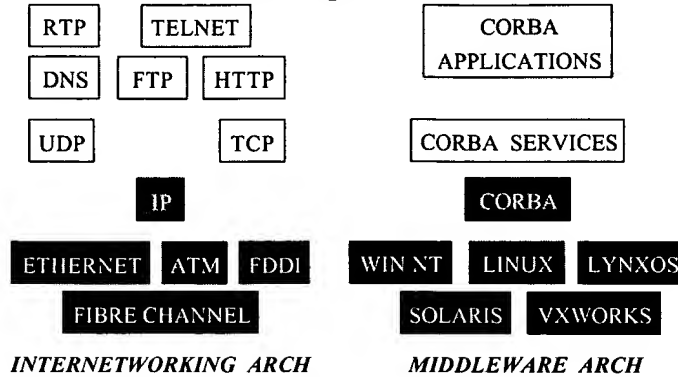
- Context
  - Adopting **COTS hardware & software** is increasingly essential for real-time mission-critical systems
- Problems
  - *Inherent* and *accidental* complexity
  - *Integration* woes
- Solution Approach
  - *Standards-based adaptive COTS middleware*

UC Irvine

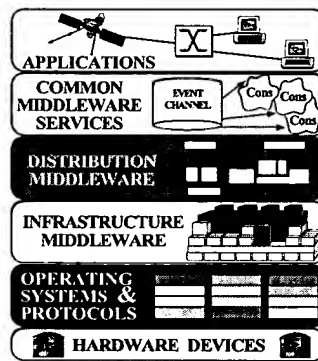


2

### Context: Levels of Abstraction in Internetworking and Middleware

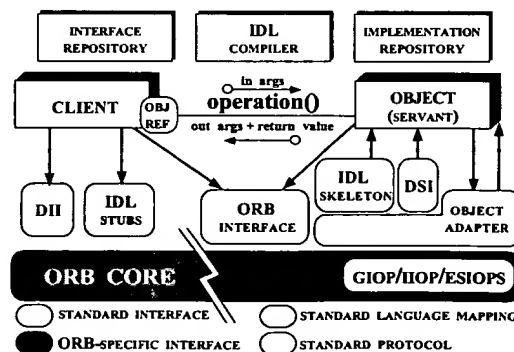


### Problem: Lack of QoS-enabled Middleware



- Many applications require QoS guarantees
  - e.g., avionics, telecom, WWW, medical, high-energy physics
- Building these applications manually is hard and inefficient
- Existing middleware doesn't support QoS effectively
  - e.g., CORBA, DCOM, DCE, Java
- Solutions must be integrated horizontally & vertically

### Candidate Solution: CORBA

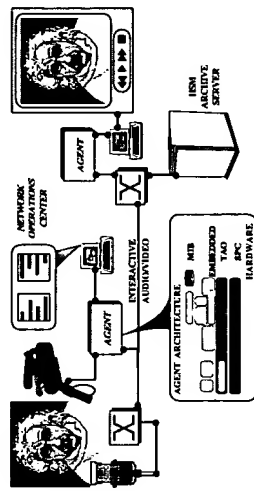


#### Goals of CORBA

- Simplify distribution by automating
  - Object location & activation
  - Parameter marshaling
  - Demultiplexing
  - Error handling
- Provide foundation for higher-level services

[www.cs.wustl.edu/~schmidt/corba.html](http://www.cs.wustl.edu/~schmidt/corba.html)

## Caveat: Requirements/Limitations of CORBA for QoS-enabled Systems



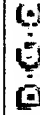
[www.cs.wustl.edu/~schmidt/RT-ORB.ps.gz](http://www.cs.wustl.edu/~schmidt/RT-ORB.ps.gz)

### Requirements

- Location transparency
- Performance transparency
- Predictability transparency
- Reliability transparency

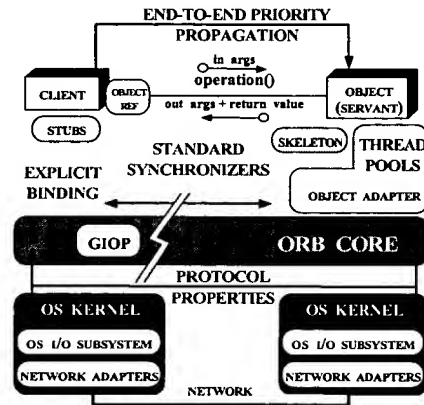
### Limitations

- Lack of QoS specifications
- Lack of QoS enforcement
- Lack of real-time programming features
- Lack of performance optimizations



UC Irvine

## Overview of the Real-time CORBA Specification



### Features

1. Portable priorities
2. End-to-end priority propagation
3. Protocol properties
4. Thread pools
5. Explicit binding
6. Standard synchronizers

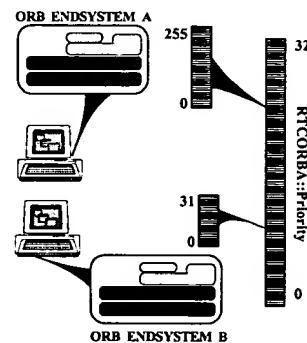
[www.cs.wustl.edu/~schmidt/orc.ps.gz](http://www.cs.wustl.edu/~schmidt/orc.ps.gz)



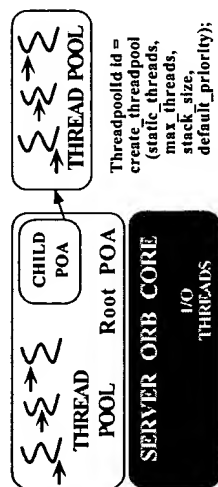
## Portable Priorities

### Features

- Designed to support heterogeneous real-time platforms
- *CORBA priorities* range from 0 → 32767
- Users can map CORBA priorities to *native OS priorities*
- No silver bullet, but rather an "enabling technique"

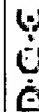


## Thread Pools



## Features

- Pre-allocate threads and thread attributes
  - Stacksize
  - Static threads
  - Maximum threads
  - Default priority
- Applicable at both the ORB and POA level



## Configurable Protocol Properties

```

interface ProtocolProperties {};

typedef struct {
    IOP::ProfileId protocol_type;
    ProtocolProperties
        orb_protocol_properties;
    ProtocolProperties
        transport_protocol_properties;
} Protocol;
typedef sequence <Protocol> ProtocolList;

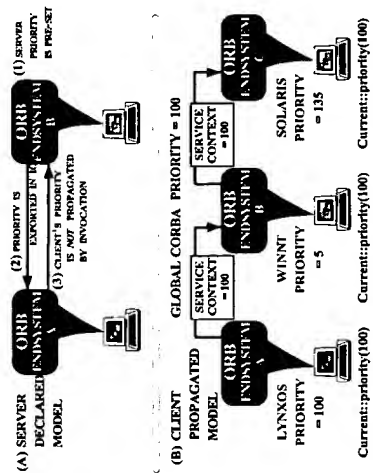
interface TCPProtocolProperties
: ProtocolProperties
{
    attribute long send_buffer_size;
    attribute long rcv_buffer_size;
    attribute boolean keep_alive;
    attribute boolean dont_route;
    attribute boolean no_delay;
};
  
```

## Features

- Select and configure communication protocols
  - e.g., TCP socket options
- Supports *ORB protocol* and *transport protocol* configuration
- Ordering in ProtocolList indicates preferences



## End-to-End Priority Propagation



www.cs.wustl.edu/~schmidt/RT-ORB-std-new.pdf.gz

## Features

- Client priorities can propagate end-to-end
- Servers can also declare priority



## Explicit Binding

\_validate\_connection (out CORBA::PolicyList inconsistent\_policies);

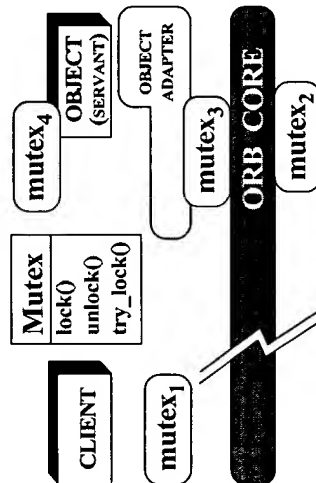


### Features

- Enables pre-establishment of connections
  - Priority-banded connections
  - Private connections
  - Protocol policies



## Standard Synchronizers

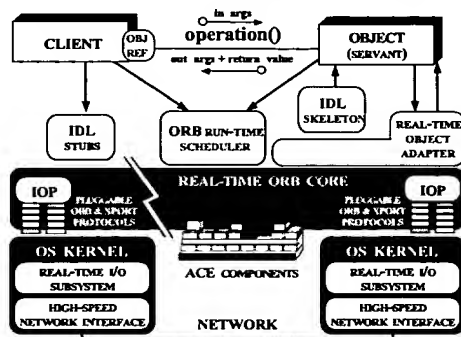


### Features

- A portable Mutex API
  - e.g., lock, unlock, try\_lock
- Necessary to ensure consistency between ORB and application synchronizers
- Locality constrained



## Our Approach: The ACE ORB (TAO)



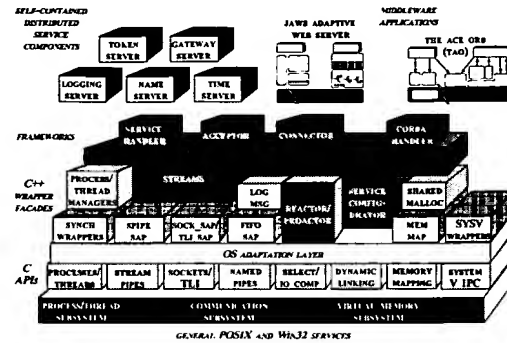
### TAO Overview →

- An open-source, standards-based, real-time, high-performance CORBA ORB
- Runs on POSIX/UNIX, Win32, & RTOS platforms
  - e.g., VxWorks, Chorus, LynxOS
- Leverages ACE

[www.cs.wustl.edu/~schmidt/TAO.html](http://www.cs.wustl.edu/~schmidt/TAO.html)



## The ADAPTIVE Communication Environment (ACE)



### ACE Overview →

- A concurrent OO networking framework
- Available in C++ and Java
- Ported to POSIX, Win32, and RTOSs

### Related work →

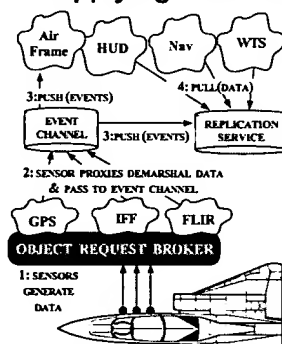
- x-Kernel
- SysV STREAMS

[www.cs.wustl.edu/~schmidt/ACE.html](http://www.cs.wustl.edu/~schmidt/ACE.html)

## ACE and TAO Statistics

- Over 50 person-years of effort
  - ACE > 200,000 LOC
  - TAO > 200,000 LOC
  - TAO IDL compiler > 130,000 LOC
  - TAO CORBA Object Services > 150,000 LOC
- Ported to UNIX, Win32, MVS, and RTOS platforms
- Large user community
  - [~schmidt/ACE-users.html](http://~schmidt/ACE-users.html)
- Currently used by dozens of companies
  - Bellcore, BBN, Boeing, Ericsson, Hughes, Kodak, Lockheed, Lucent, Motorola, Nokia, Nortel, Raytheon, SAIC, Siemens, etc.
- Supported commercially
  - ACE → [www.riverace.com](http://www.riverace.com)
  - TAO → [www.theaceorb.com](http://www.theaceorb.com)

## Applying TAO to Avionics Mission Computing



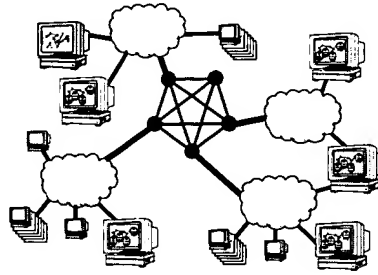
### Domain Challenges

- Deterministic & statistical real-time deadlines
- Periodic & aperiodic processing
- COTS and open systems
- Reusable components
- Support platform upgrades

[www.cs.wustl.edu/~schmidt/TAO-boeing.html](http://www.cs.wustl.edu/~schmidt/TAO-boeing.html)

[www.cs.wustl.edu/~schmidt/JSAC-98.ps.gz](http://www.cs.wustl.edu/~schmidt/JSAC-98.ps.gz)

## Applying TAO to Distributed Interactive Simulations



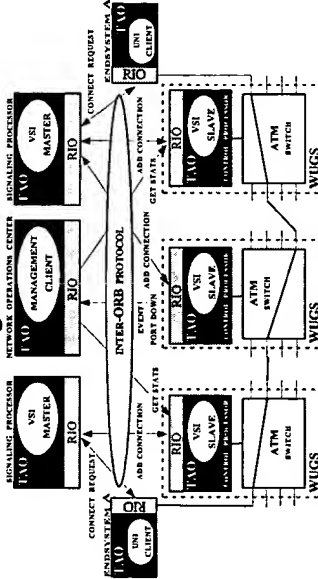
[www.cs.wustl.edu/~schmidt/Words99.ps.gz](http://www.cs.wustl.edu/~schmidt/Words99.ps.gz)

### Domain Challenges

- High scalability and group communication
- High throughput and low latency
- "Interactive" real-time
- Multi-platform

[hlsdc.dmsi.mil/RTISUP/hla\\_soft/hla\\_soft.htm](http://hlsdc.dmsi.mil/RTISUP/hla_soft/hla_soft.htm)

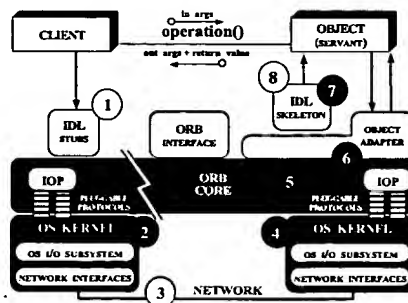
## Applying TAO to Real-time Switch Management



### Domain Challenges

- High-speed (20 Gbps) ATM switches w/embedded controllers
- Low-latency and statistical real-time deadlines
- COTS infrastructure, open systems, and small footprint

## Optimization Challenges for QoS-enabled ORBs

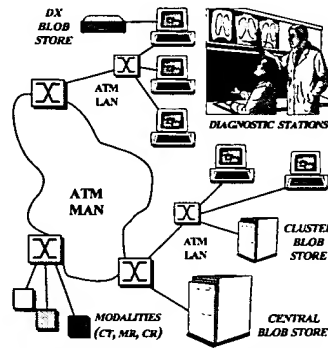


### Key Challenges

- Alleviate priority inversion and non-determinism
- Reduce demultiplexing latency/jitter
- Ensure protocol flexibility
- Specify QoS requirements
- Schedule operations
- Eliminate (de)marshaling overhead
- Minimize footprint



## Problem: Optimizing Complex Software



[www.cs.wustl.edu/~schmidt/JSAC-99.ps.gz](http://www.cs.wustl.edu/~schmidt/JSAC-99.ps.gz)

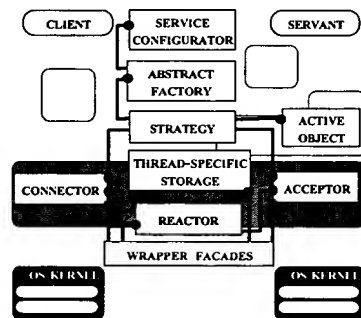
### Common Problems →

- Optimizing complex software is hard
- Small "mistakes" can be costly

### Solution Approach (Iterative) →

- Pinpoint overhead via *white-box* metrics
  - e.g., Quantify and VMetro
- Apply patterns and framework components
- Revalidate via *white-box* and *black-box* metrics

## Solution 1: Patterns and Framework Components



[www.cs.wustl.edu/~schmidt/ORB-patterns.ps.gz](http://www.cs.wustl.edu/~schmidt/ORB-patterns.ps.gz)

### Definitions

- *Pattern*
  - A solution to a problem in a context
- *Framework*
  - A "semi-complete" application built with components
- *Components*
  - Self-contained, "pluggable" ADTs

## Solution 2: ORB Optimization Principle Patterns

### Definition

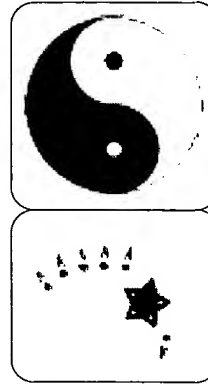
- *Optimization principle patterns* document rules for avoiding common design and implementation problems that can degrade the efficiency, scalability, and predictability of complex systems

### Optimization Principle Patterns Used in TAO

#	Optimization Principle Pattern
1	Optimize for the common case
2	Remove gratuitous waste
3	Replace inefficient general-purpose functions with efficient special-purpose ones
4	Shift computation in time, e.g., precompute
5	Store redundant state to speed-up expensive operations
6	Pass hints between layers and components
7	Don't be tied to reference implementations/models
8	Use efficient/predictable data structures

## Lessons Learned Developing QoS-enabled ORBs

- Avoid dynamic connection management
- Minimize dynamic memory management and data copying
- Avoid multiplexing connections for different priority threads
- Avoid complex concurrency models
- Integrate ORB with OS and I/O subsystem and avoid reimplementing OS mechanisms
- Guide ORB design by empirical benchmarks and patterns



## Concluding Remarks

- Researchers and developers of distributed, real-time applications confront many common challenges
  - e.g., service initialization and distribution, error handling, flow control, scheduling, event demultiplexing, concurrency control, persistence, fault tolerance
- Successful researchers and developers apply *patterns*, *frameworks*, and *components* to resolve these challenges
- Careful application of patterns can yield efficient, predictable, scalable, *and* flexible middleware
  - i.e., middleware performance is largely an “implementation detail”
- Next-generation ORBs will be highly QoS-enabled, though many research challenges remain

## Web URLs for Additional Information

- These slides: [~schmidt/TAO4.ps.gz](#)
- More information on CORBA: [~schmidt/corba.html](#)
- More info on ACE: [~schmidt/ACE.html](#)
- More info on TAO: [~schmidt/TAO.html](#)
- TAO Event Channel: [~schmidt/JSAC-98.ps.gz](#)
- TAO static scheduling: [~schmidt/TAO.ps.gz](#)
- TAO dynamic scheduling: [~schmidt/dynamic.ps.gz](#)
- ORB Endsystem Architecture: [~schmidt/RIO.ps.gz](#)
- Pluggable protocols: [~schmidt/pluggable\\_protocols.ps.gz](#)

**Web URLs for Additional Information (cont'd)**

- Network monitoring, visualization, & control: [~schmidt/NMVC.html](#)
- Performance Measurements:
  - Demuxing latency: [~schmidt/COOTS-99.ps.gz](#)
  - SII throughput: [~schmidt/SIGCOMM-96.ps.gz](#)
  - DII throughput: [~schmidt/GLOBECOM-96.ps.gz](#)
  - ORB latency & scalability: [~schmidt/ieee\\_tc-97.ps.gz](#)
  - I/O optimizations: [~schmidt/JSAC-99.ps.gz](#)
  - Concurrency and connection models: [~schmidt/RT-perf.ps.gz](#)
  - RTOS/ORB benchmarks:
    - [~schmidt/RT-OS.ps.gz](#)
    - [~schmidt/words-99.ps.gz](#)

# Patterns and Performance of Real-time Object Request Brokers

**Douglas C. Schmidt**

Associate Professor  
schmidt@uci.edu

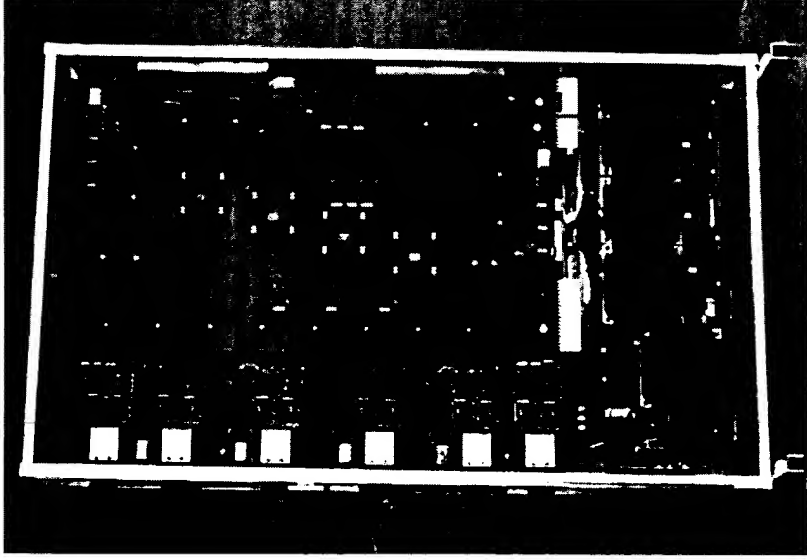
Elec. & Comp. Eng. Dept.  
University of California, Irvine  
www.ece.uci.edu/~schmidt/ (949) 824-1901



## **Sponsors**

NSF, DARPA, ATD, BBN, Boeing, Cisco, Comverse, GDIS, Experian, Global MT,  
Hughes, Kodak, Krones, Lockheed, Lucent, Microsoft, Mitre, Motorola, NASA, Nokia,  
Nortel, OCl, Oresis, OTI, Raytheon, SAIC, Siemens SCR, Siemens MED, Siemens  
ZT, Sprint, Telcordia, USENIX

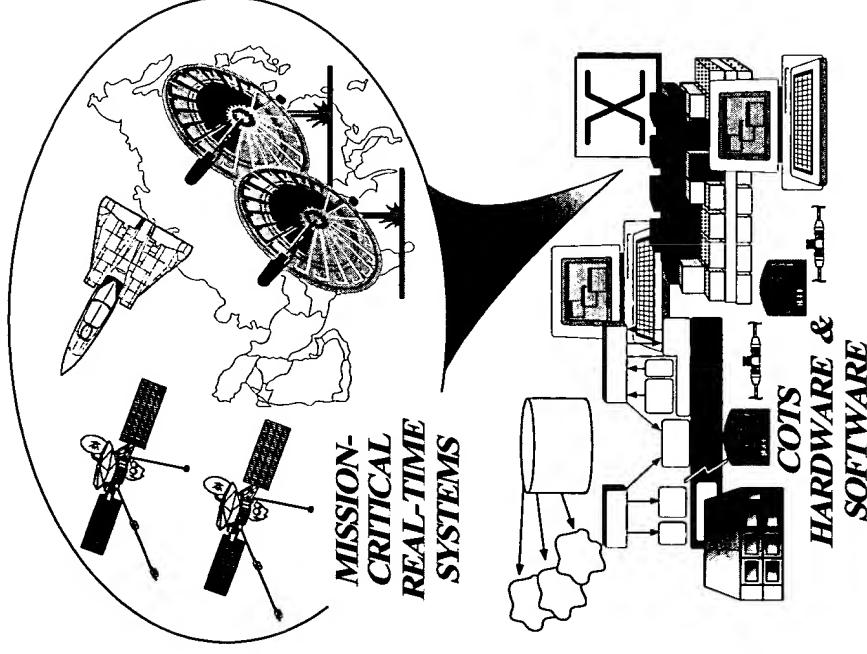
## Motivation: the QoS-enabled Software Crisis



[www.arl.wustl.edu/arl/](http://www.arl.wustl.edu/arl/)

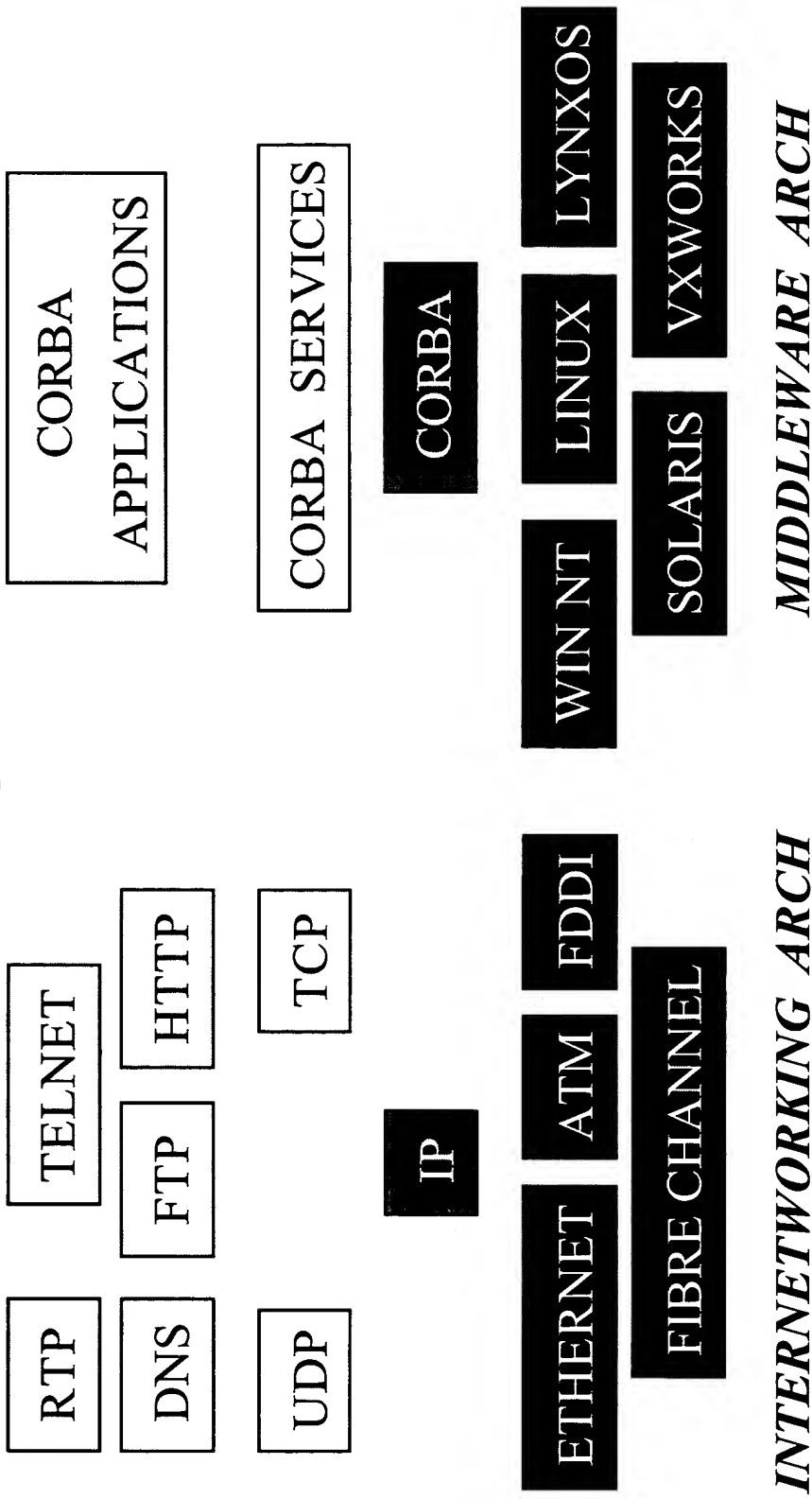
- Symptoms
  - Communication *hardware* gets smaller, faster, cheaper
  - Communication *software* gets larger, slower, more expensive
- Culprits
  - *Inherent* and *accidental* complexity
- Solution Approach
  - *Standards-based COTS Hardware & Software*

## Problem: the COTS Hardware & Software Crisis

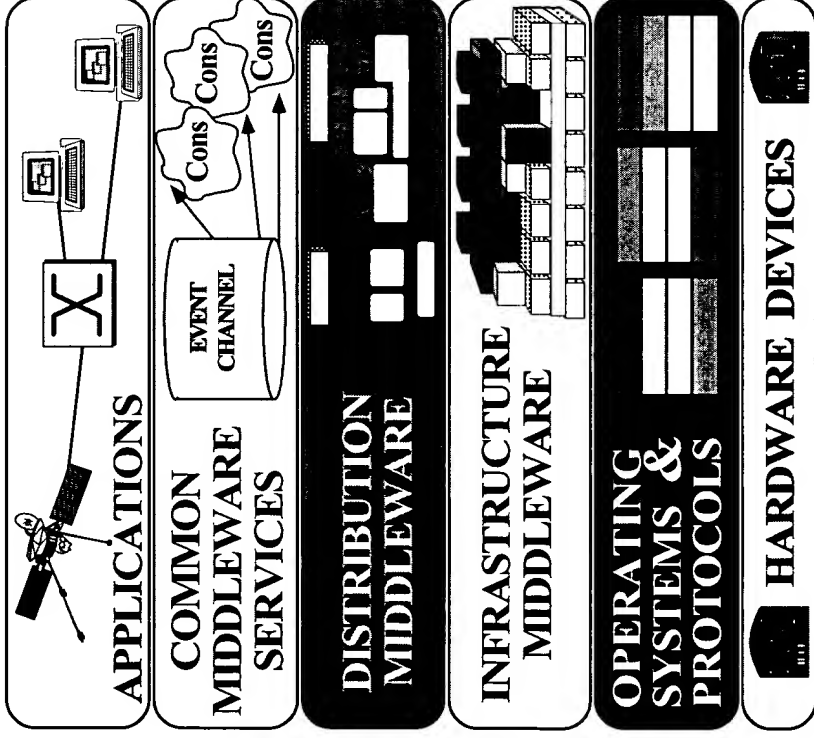


- Context
  - Adopting *COTS hardware & software* is increasingly essential for real-time mission-critical systems
- Problems
  - *Inherent* and *accidental* complexity
  - *Integration* woes
- Solution Approach
  - *Standards-based adaptive COTS middleware*

## Context: Levels of Abstraction in Internetworking and Middleware



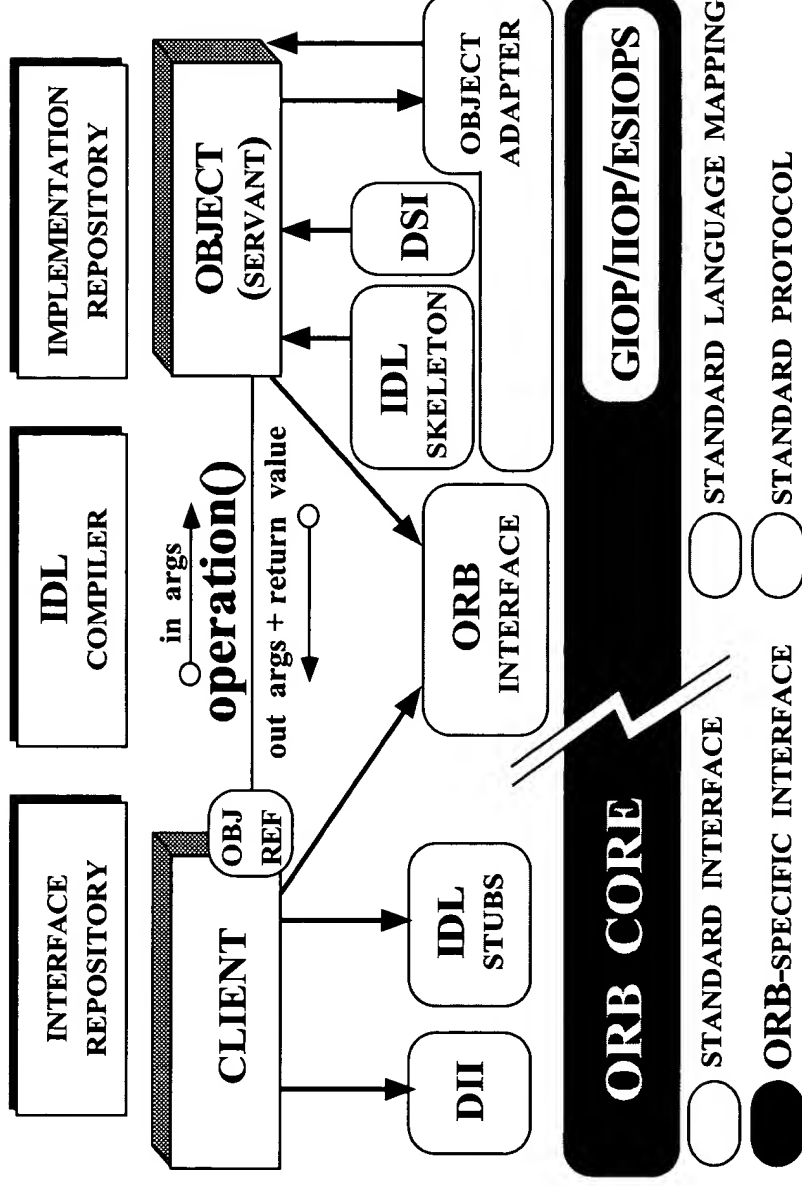
## Problem: Lack of QoS-enabled Middleware



- Many applications require QoS guarantees
  - e.g., avionics, telecom, WWW, medical, high-energy physics
- Building these applications manually is hard and inefficient
- Existing middleware doesn't support QoS effectively
  - e.g., CORBA, DCOM, DCE, Java
- Solutions must be integrated horizontally & vertically



## Candidate Solution: CORBA

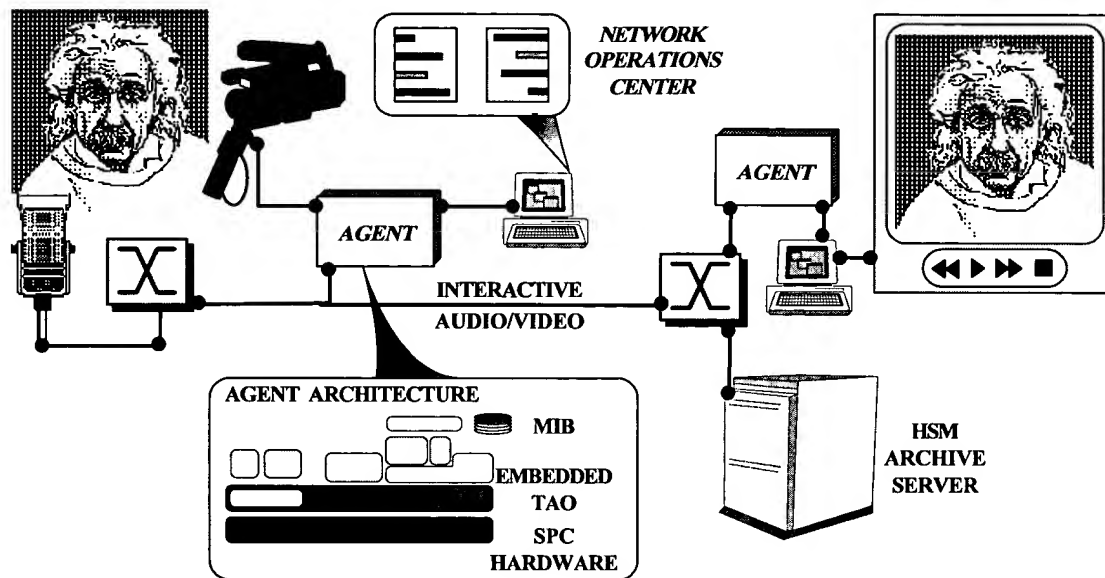


[www.cs.wustl.edu/~schmidt/corba.html](http://www.cs.wustl.edu/~schmidt/corba.html)

### Goals of CORBA

- Simplify distribution by automating
  - Object location & activation
  - Parameter marshaling
  - Demultiplexing
  - Error handling
- Provide foundation for higher-level services

## Caveat: Requirements/Limitations of CORBA for QoS-enabled Systems



[www.cs.wustl.edu/~schmidt/RT-ORB.ps.gz](http://www.cs.wustl.edu/~schmidt/RT-ORB.ps.gz)

### Requirements

- Location transparency
- Performance transparency
- Predictability transparency
- Reliability transparency

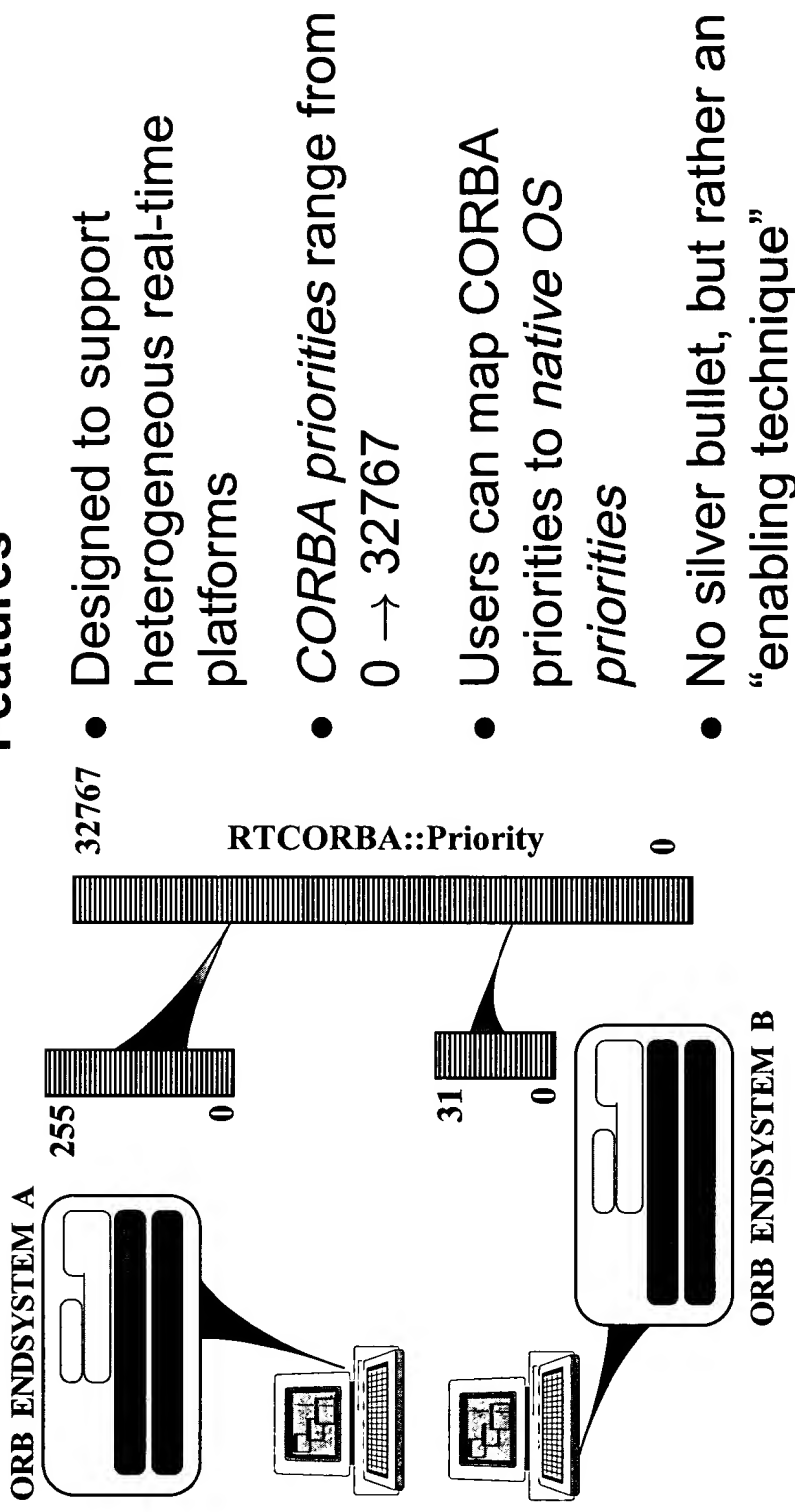
### Limitations

- Lack of QoS specifications
- Lack of QoS enforcement
- Lack of real-time programming features
- Lack of performance optimizations

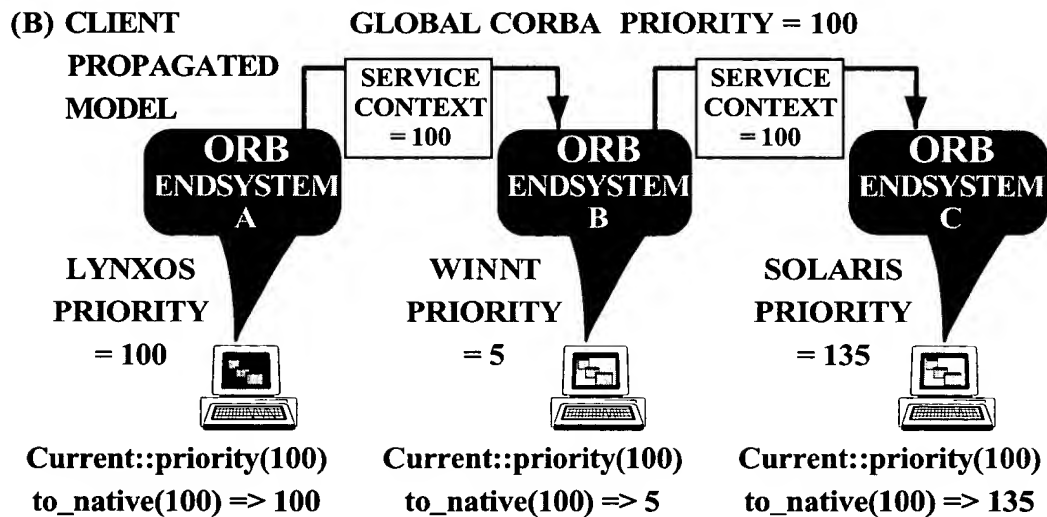
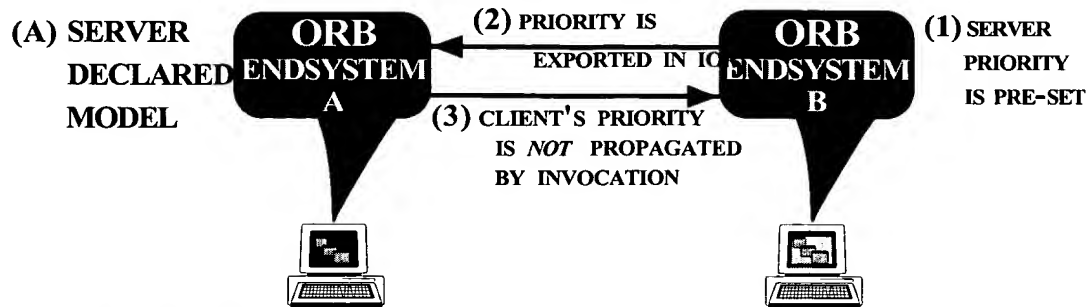


## Portable Priorities

### Features



## End-to-End Priority Propagation



[www.cs.wustl.edu/~schmidt/RT-ORB-standard-new.pdf.gz](http://www.cs.wustl.edu/~schmidt/RT-ORB-standard-new.pdf.gz)

## Features

- Client priorities can propagate end-to-end
- Servers can also declare priority

## Configurable Protocol Properties

```
interface ProtocolProperties {};
```

### Features

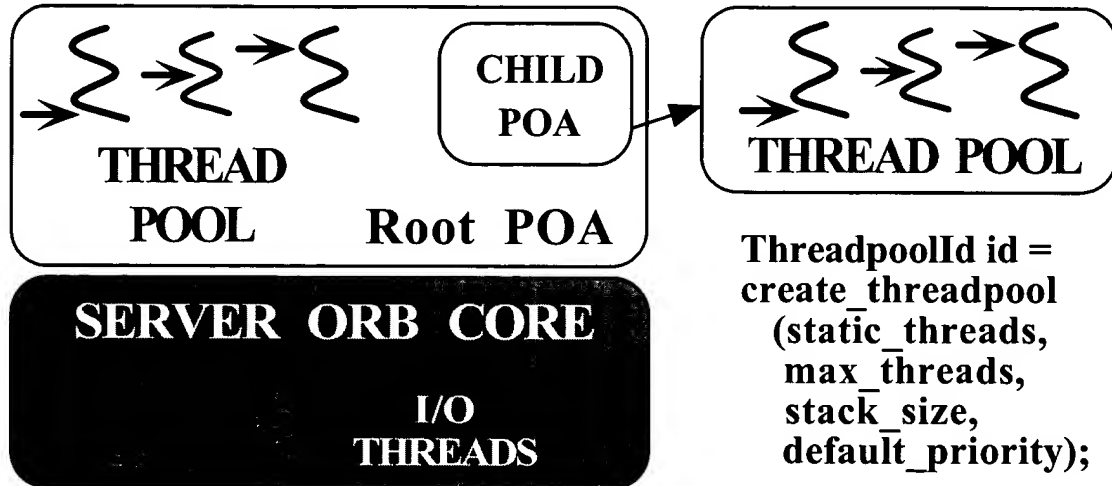
```
typedef struct {
    IOP::ProfileId protocol_type;
    ProtocolProperties
        orb_protocol_properties;
    ProtocolProperties
        transport_protocol_properties;
} Protocol;
typedef sequence <Protocol> ProtocolList;
```

- Select and configure communication protocols
  - e.g., TCP socket options

```
interface TCPProtocolProperties
: ProtocolProperties
{
    attribute long send_buffer_size;
    attribute long recv_buffer_size;
    attribute boolean keep_alive;
    attribute boolean dont_route;
    attribute boolean no_delay;
};
```

- Supports *ORB protocol* and *transport protocol* configuration
- Ordering in ProtocolList indicates preferences

## Thread Pools

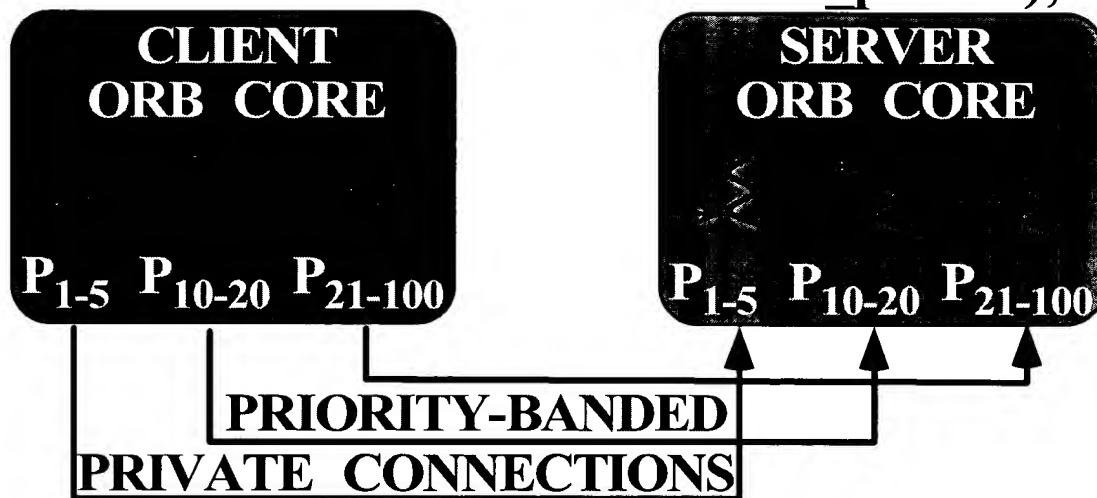


## Features

- Pre-allocate threads and thread attributes
  - Stacksize
  - Static threads
  - Maximum threads
  - Default priority
- Applicable at both the ORB and POA level

## Explicit Binding

`_validate_connection` (out `CORBA::PolicyList`  
`inconsistent_policies`);

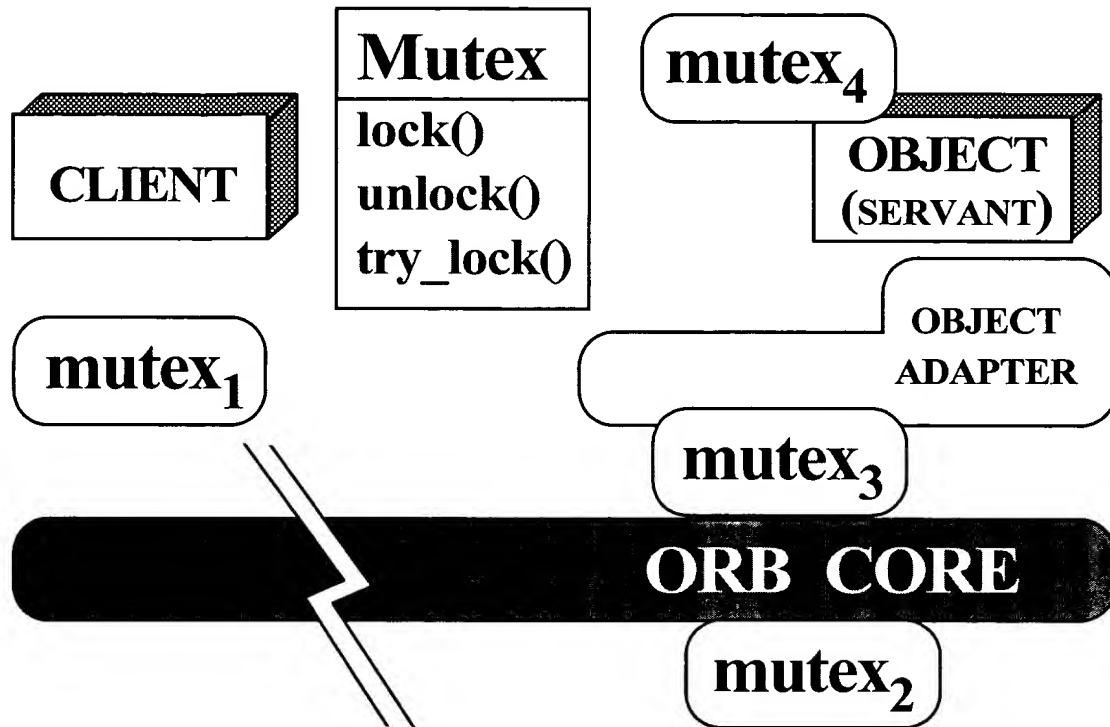


### Features

- Enables pre-establishment of connections
  - Priority-banded connections
  - Private connections
  - Protocol policies



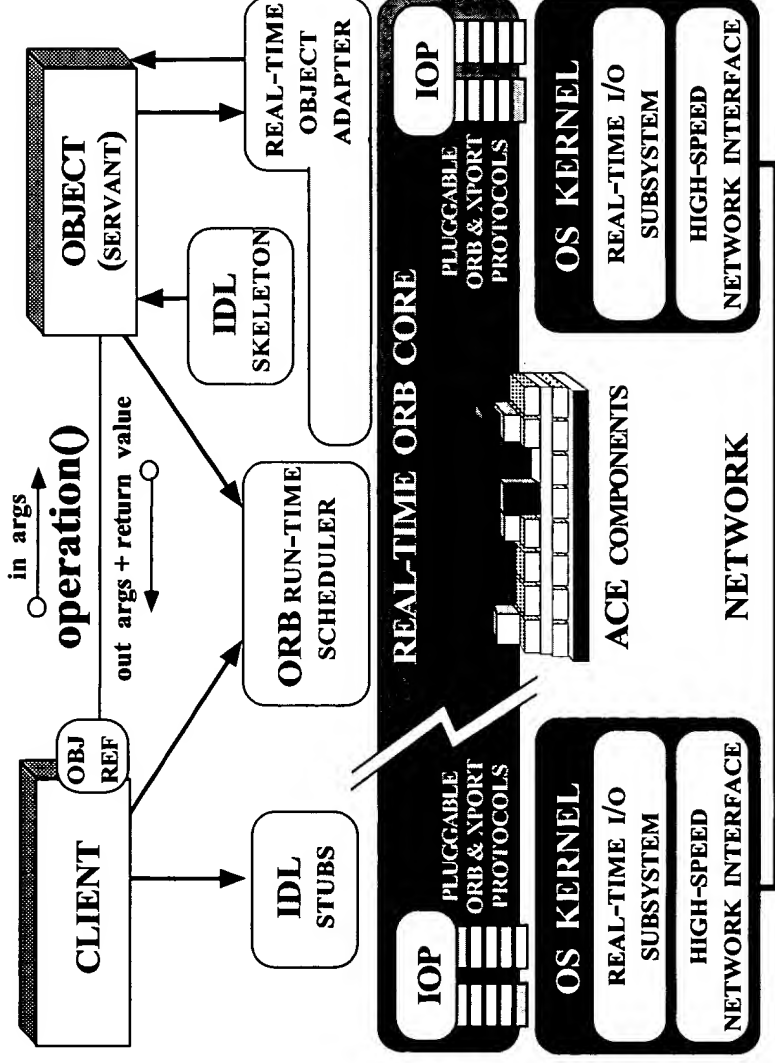
## Standard Synchronizers



### Features

- A portable Mutex API
  - *e.g.*, `lock`, `unlock`, `try_lock`
- Necessary to ensure consistency between ORB and application synchronizers
- Locality constrained

## Our Approach: The ACE ORB (TAO)

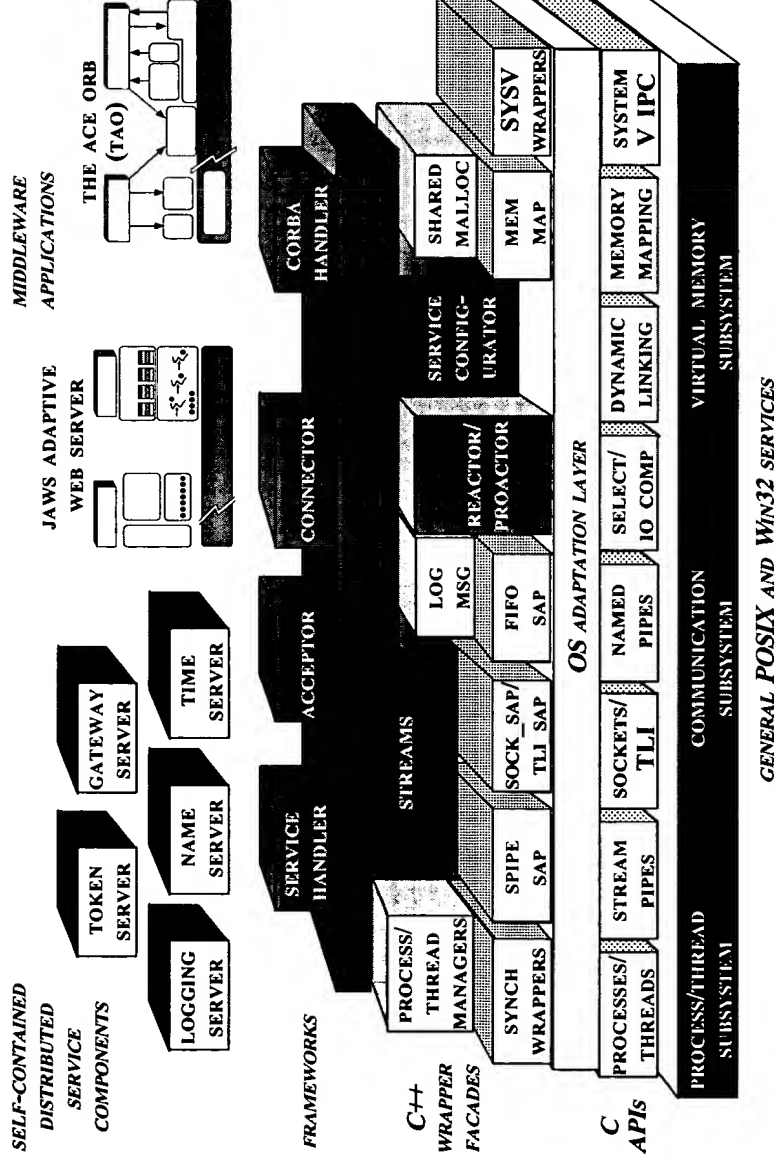


### TAO Overview →

- An open-source, standards-based, real-time, high-performance CORBA ORB
- Runs on POSIX/UNIX, Win32, & RTOS platforms
  - e.g., VxWorks, Chorus, LynxOS
- Leverages ACE

[www.cs.wustl.edu/~schmidt/TAO.html](http://www.cs.wustl.edu/~schmidt/TAO.html)

# The ADAPTIVE Communication Environment (ACE)



## ACE Overview →

- A concurrent OO networking framework
- Available in C++ and Java
- Ported to POSIX, Win32, and RTOSS

## Related work →

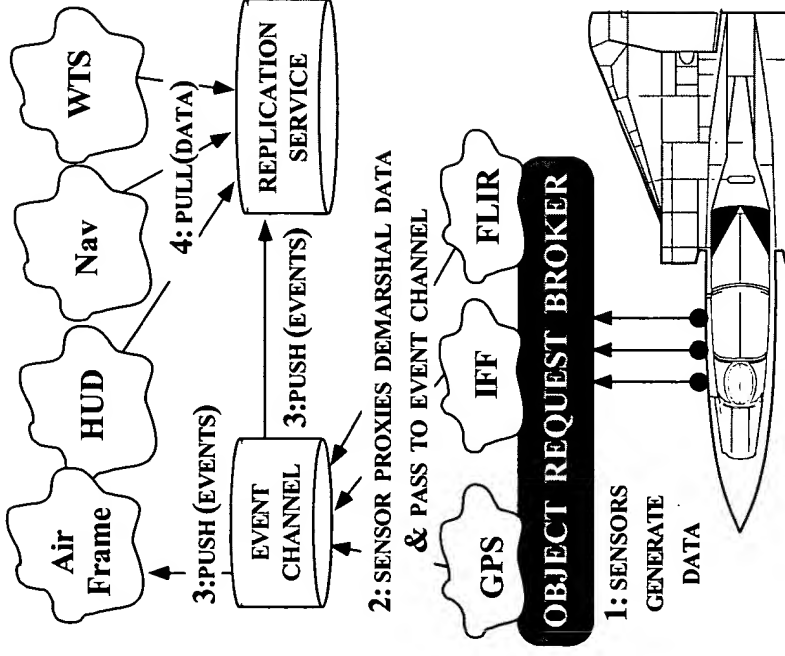
- x-Kernel
- SysV STREAMS

[www.cs.wustl.edu/~schmidt/ACE.html](http://www.cs.wustl.edu/~schmidt/ACE.html)

## ACE and TAO Statistics

- Over 50 person-years of effort
  - ACE > 200,000 LOC
  - TAO > 200,000 LOC
  - TAO IDL compiler > 130,000 LOC
  - TAO CORBA Object Services > 150,000 LOC
- Ported to UNIX, Win32, MVS, and RTOS platforms
- Supported commercially
- Large user community
  - ACE → [www.riverace.com](http://www.riverace.com)
  - TAO → [www.theaceorb.com](http://www.theaceorb.com)
- Currently used by dozens of companies
  - Bellcore, BBN, Boeing, Ericsson, Hughes, Kodak, Lockheed, Lucent, Motorola, Nokia, Nortel, Raytheon, SAIC, Siemens, etc.

# Applying TAO to Avionics Mission Computing

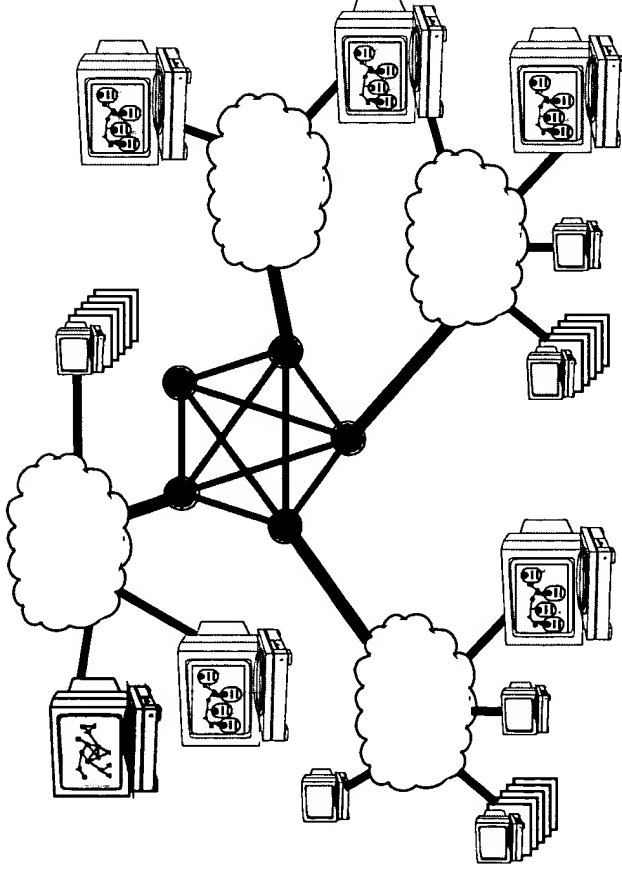


- ## Domain Challenges
- Deterministic & statistical real-time deadlines
  - Periodic & aperiodic processing
  - COTS and open systems
  - Reusable components
  - Support platform upgrades

[www.cs.wustl.edu/~schmidt/TAO-boeing.html](http://www.cs.wustl.edu/~schmidt/TAO-boeing.html)

[www.cs.wustl.edu/~schmidt/JSAC-98.ps.gz](http://www.cs.wustl.edu/~schmidt/JSAC-98.ps.gz)

## Applying TAO to Distributed Interactive Simulations



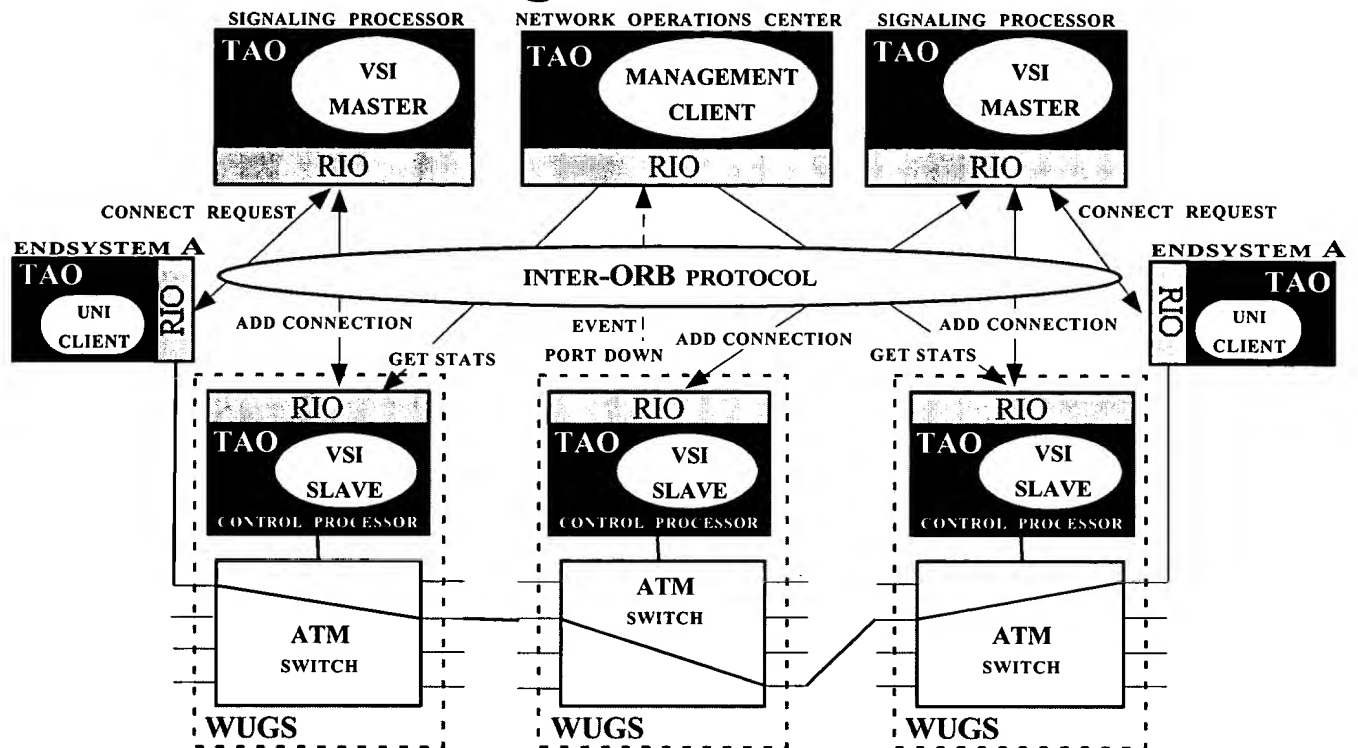
[www.cs.wustl.edu/~schmidt/  
Words99.ps.gz](http://www.cs.wustl.edu/~schmidt/Words99.ps.gz)

### Domain Challenges

- High scalability and group communication
- High throughput and low latency
- “Interactive” real-time
- Multi-platform

[hlasdc.dmsoc.mil/RTISUP/hla\\_soft/  
hla\\_soft.htm](http://hlasdc.dmsoc.mil/RTISUP/hla_soft/hla_soft.htm)

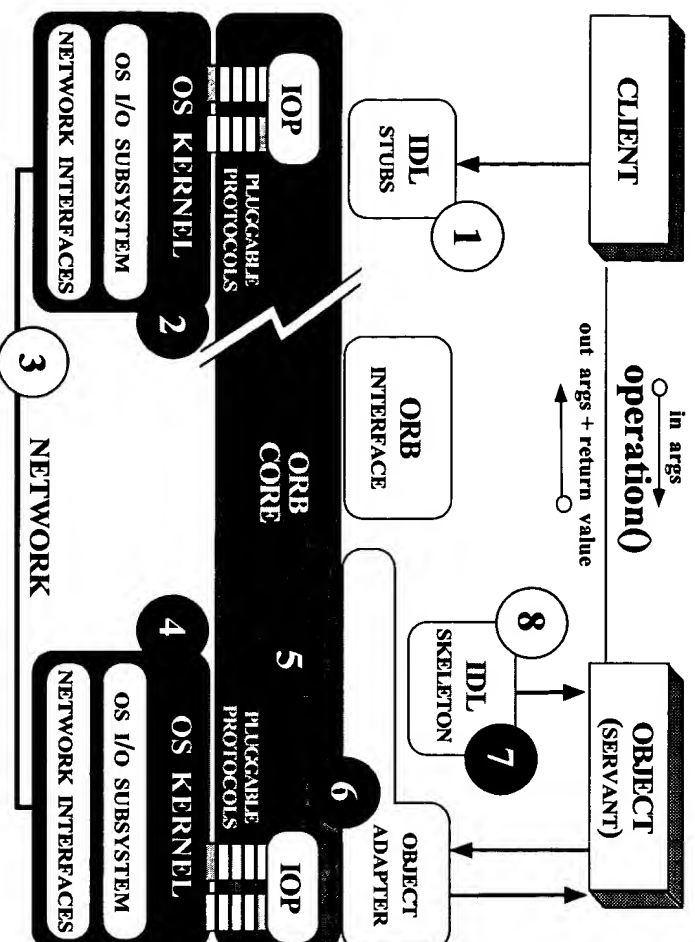
# Applying TAO to Embedded Network Element Management and Control



## Domain Challenges

- High-speed (20 Gbps) ATM switches w/embedded controllers
- Low-latency and statistical real-time deadlines
- COTS infrastructure, standards-based open systems, and small footprint

## Optimization Challenges for QoS-enabled ORBs



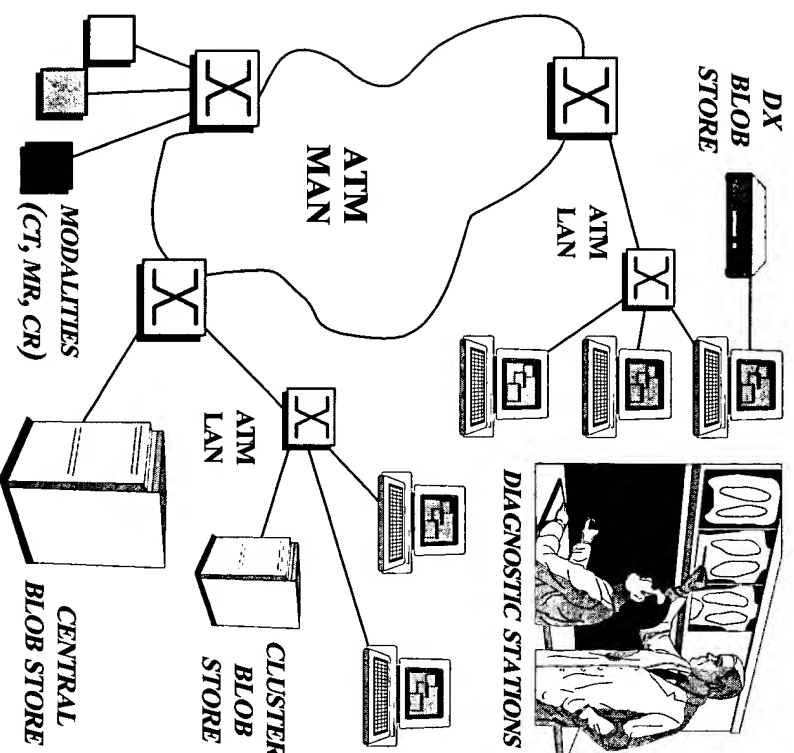
- 1) CLIENT MARSHALING
- 2) CLIENT PROTOCOL
- 3) NETWORK LATENCY
- 4) SERVER PROTOCOL
- 5) THREAD DISPATCHING
- 6) REQUEST DEMUXING
- 7) OPERATION DEMUXING
- 8) SERVANT DEMARSHALING

### Key Challenges

- Alleviate priority inversion and non-determinism
- Reduce demultiplexing latency/jitter
- Ensure protocol flexibility
- Specify QoS requirements
- Schedule operations
- Eliminate (de)marshaling overhead
- Minimize footprint



## Problem: Optimizing Complex Software



### Common Problems →

- Optimizing complex software is hard
- Small “mistakes” can be costly

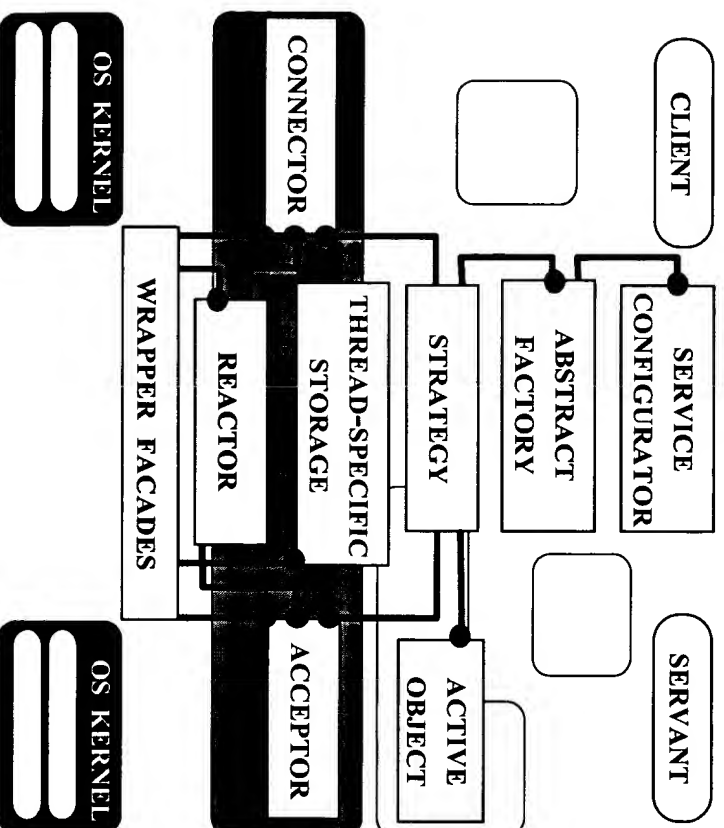
### Solution Approach (Iterative) →

- Pinpoint overhead via *white-box* metrics
  - e.g., Quantify and VMETRO
- Apply patterns and framework components
- Revalidate via *white-box* and *black-box* metrics

[www.cs.wustl.edu/~schmidt/](http://www.cs.wustl.edu/~schmidt/)

JSAC-99.ps.gz

## Solution 1: Patterns and Framework Components



### Definitions

- *Pattern*
  - A solution to a problem in a context
- *Framework*
  - A “semi-complete” application built with components
- *Components*
  - Self-contained, “pluggable” ADTs

[www.cs.wustl.edu/~schmidt/ORB-patterns.ps.gz](http://www.cs.wustl.edu/~schmidt/ORB-patterns.ps.gz)

## Solution 2: ORB Optimization Principle Patterns

### Definition

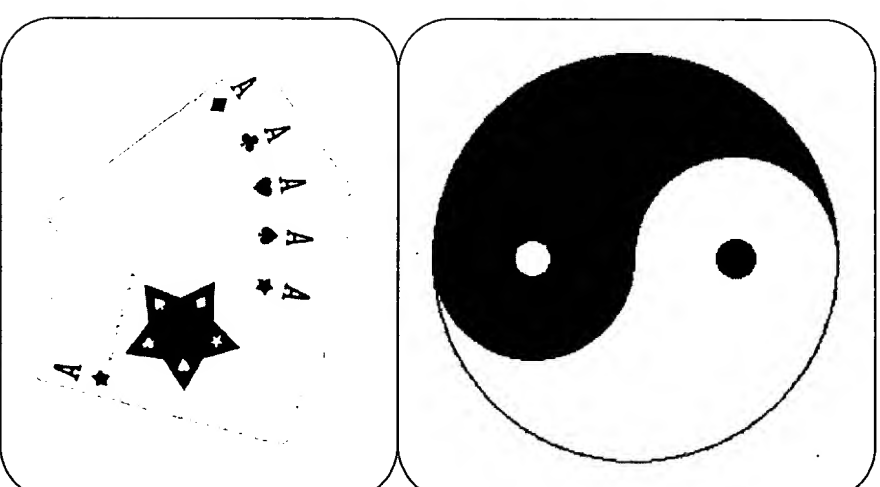
- ***Optimization principle patterns*** document rules for avoiding common design and implementation problems that can degrade the efficiency, scalability, and predictability of complex systems

### Optimization Principle Patterns Used in TAO

#	Optimization Principle Pattern
1	Optimize for the common case
2	Remove gratuitous waste
3	Replace inefficient general-purpose functions with efficient special-purpose ones
4	Shift computation in time, e.g., precompute
5	Store redundant state to speed-up expensive operations
6	Pass hints between layers and components
7	Don't be tied to reference implementations/models
8	Use efficient/predictable data structures

## Lessons Learned Developing QoS-enabled ORBs

- Avoid dynamic connection management
- Minimize dynamic memory management and data copying
- Avoid multiplexing connections for different priority threads
- Avoid complex concurrency models
- Integrate ORB with OS and I/O subsystem and avoid reimplementing OS mechanisms
- Guide ORB design by empirical benchmarks and patterns



---

## Concluding Remarks

- Researchers and developers of distributed, real-time applications confront many common challenges
    - *e.g.*, service initialization and distribution, error handling, flow control, scheduling, event demultiplexing, concurrency control, persistence, fault tolerance
  - Successful researchers and developers apply *patterns*, *frameworks*, and *components* to resolve these challenges
  - Careful application of patterns can yield efficient, predictable, scalable, *and* flexible middleware
    - *i.e.*, middleware performance is largely an “implementation detail”
  - Next-generation ORBs will be highly QoS-enabled, though many research challenges remain
-

---

## Web URLs for Additional Information

- These slides: `~schmidt/TAO4.ps.gz`
- More information on CORBA: `~schmidt/corba.html`
- More info on ACE: `~schmidt/ACE.html`
- More info on TAO: `~schmidt/TAO.html`
- TAO Event Channel: `~schmidt/JSAC-98.ps.gz`
- TAO static scheduling: `~schmidt/TAO.ps.gz`
- TAO dynamic scheduling: `~schmidt/dynamic.ps.gz`
- ORB Endsystem Architecture: `~schmidt/RIO.ps.gz`
- Pluggable protocols: `~schmidt/pluggable_protocols.ps.gz`

---

## Web URLs for Additional Information (cont'd)

---

- Network monitoring, visualization, & control: `~schmidt/NMVC.html`
- Performance Measurements:
  - Demuxing latency: `~schmidt/COOTS-99.ps.gz`
  - SII throughput: `~schmidt/SIGCOMM-96.ps.gz`
  - DII throughput: `~schmidt/GLOBECOM-96.ps.gz`
  - ORB latency & scalability: `~schmidt/ieee-tc-97.ps.gz`
  - IIOF optimizations: `~schmidt/JSAC-99.ps.gz`
  - Concurrency and connection models: `~schmidt/RT-perf.ps.gz`
  - RTOS/ORB benchmarks:
    - `~schmidt/RT-OS.ps.gz`
    - `~schmidt/words-99.ps.gz`